Nonlinear Dyn (2017) 88:1733–1747
DOI 10.1007/s11071-017-3342-3

ORIGINAL PAPER

Egalitarian versus prioritarian approach in multiple task motion planning for nonholonomic systems

Adam Ratajczak

Received: 1 July 2016 / Accepted: 6 January 2017 / Published online: 19 January 2017

Abstract In this paper, two different concepts of multiple task motion planning algorithm for nonholonomic systems are considered. The egalitarian approach treats all the tasks equivalently and tries to solve all the tasks simultaneously. In contrast, the prioritarian approach arranges the tasks with decreasing priorities in such a way that the solution of the lower order task should not influence the solution of the higher order task. This paper contains the derivation of the egalitarian motion planning algorithm and the prioritarian motion planning algorithm. Moreover, the definitions of the three types of basic subtasks are also included. The efficiency of the egalitarian and prioritarian algorithms is presented with the simulation of the nonholonomic model of the unmanned surface vessel. The simulation results provide the data to perform a comparison of the egalitarian versus the prioritarian approach.

Keywords Nonholonomic system · Jacobian motion planning · Multiple task · Egalitarian algorithm · Prioritarian algorithm · Unmanned surface vessel

A. Ratajczak (✉)
Department of Control Systems and Mechatronics,
Wrocław University of Science and Technology,
Janiszewski str. 11/17, 50-372 Wrocław, Poland
e-mail: adam.ratajczak@pwr.edu.pl

1 Introduction

A solution of the motion planning problem in nonholonomic systems provides the control function which drives the system from an initial to a desired point. The multiple task motion planning problem is an extension of the classical motion planning problem. The multiple task motion planning algorithm solves the proper motion planning task along with one or more additional tasks, named subtasks. These subtasks may be diverse and may depend on the control, state space or task space variables.

The idea of multiple task motion planning in robotics comes from the theory for redundant manipulators. According to [6], there are several ways to solve the inverse kinematics problem for manipulators with joint redundancy. In general, the inverse kinematics problem for redundant manipulators has more than one solution, and the main issue is how to choose the desired one. On the other hand, this fact could be used to solve some additional tasks beyond the inverse kinematics problem. Two main techniques have been developed to solve the inverse kinematics problem with additional subtasks. The first one based on the extended Jacobian was proposed in [2] and revised in [4]. This technique uses specific extending functions which produce the square extended Jacobian matrix, finally inverted to obtain a solution of the inverse kinematics problem. A very similar approach is presented in [16], where the augmented Jacobian is defined. This approach, either extended Jacobian or augmented Jacobian, treats the

main task (the proper motion planning) equivalently with additional subtasks, thus we will call it an egalitarian approach. The second approach used to solve the multiple task problem is the task priority approach [5,11]. This method arranges the subtasks in accordance with decreasing priorities. As long as the manipulator is redundant, the algorithm's design can be based on the fact that the kernel of the Jacobian matrix is non empty. This strategy results in that the solution of a task with lower priority does not influence the solution of the higher priority task. For the sake of simplicity, we will refer to that method as a prioritarian approach.

This work focusses on the egalitarian multiple task motion planning algorithm and the prioritarian multiple task motion planning algorithm for nonholonomic systems. These two algorithms are derived within the endogenous configuration space approach [19]. This approach introduces many analogies between the holonomic robotic manipulators and the nonholonomic robotics systems. Relying on this analogy, we shall introduce two multiple task motion planning algorithms. Some applications of the multiple task motion planning algorithm based on the endogenous configuration space approach have already been published. For example, the work [9] introduces the augmented Jacobian algorithm. In the paper [20], the extended Jacobian is used to approximate another Jacobian algorithm. The prioritarian approach could be found in [13,14].

The primary contribution of this work is the design of two multiple task motion planning algorithms. The paper [13] introduces the prioritarian multiple task motion planning algorithm. Contrary to [13], in the presented paper, the derivation of the prioritarian algorithm is provided in more systematic way. In addition, this paper also includes the derivation of the egalitarian multiple task motion planning algorithm. Moreover, the three types of basic subtasks are defined relaying on the theory introduced in [14]. These subtasks may be treated as a base for the construction of more complicated subtasks. The paper also presents a discussion about numerical aspects and the guidelines concerning the algorithms implementation. Additional contribution is the comparison between the egalitarian approach and the prioritarian one based on the simulation results.

A secondary contribution of this paper is the implementation of both multiple task algorithms by means of the nonparametric and higher order differential equation solver [15]. The paper [15] presents the various approaches to numerical computation of the single task motion planning algorithm. In this paper, this approaches are extended to the multiple task motion planning algorithms.

Up to now, there is no existing work which introduces the systematic derivation of the egalitarian and prioritarian motion planning algorithms. Moreover, the previous works did not discuss the numerical aspects of the implementation of both algorithms with nonparametric approach.

The remaining parts of this work is arranged in the following way. Section 2 introduces preliminaries about the endogenous configuration space. The definitions of both multiple task Jacobian motion planning algorithms are presented in Sect. 3. In Sect. 4 we define three types of subtasks. Computer simulation results and the comparison of the egalitarian and the prioritarian algorithm are collected in Sect. 5. Section 6 contains conclusions.

2 Endogenous configuration space

The endogenous configuration space approach [19] will be adopted to a control affine system of the form

$$\begin{cases} \frac{\mathrm{d}q}{\mathrm{d}t} = f(q) + G(q)u = f(q) + \sum_{i=1}^{m} g_i(q)u_i, \\ y = k(q), \end{cases} \tag{1}$$

where $q \in \mathbf{R}^n$ is a state space vector, $u \in \mathbf{R}^m$ represents control vector and $y \in \mathbf{R}^r$ is a vector of coordinates in the task space. $G(q)$ is a $n \times m$ control matrix, $f(q)$ is a drift term and $k(q)$ denotes an output function. We assume that all the vectors and functions appearing in (1) are smooth. The admissible control functions $u(t)$ belong to the control space $\mathcal{U}$ named the endogenous configuration space. The endogenous configuration space $\mathcal{U} \subset L_m^2[0, T]$ is a space of Lebesque square integrable functions defined on the time interval $t \in [0, T]$. Moreover, the space $L_m^2[0, T]$ is a Hilbert space with the inner product

$$\langle u_1(\cdot), u_2(\cdot) \rangle = \int_0^T u_1^{\mathsf{T}}(t) u_2(t)\, \mathrm{d}t, \tag{2}$$

and the norm

$$\|u(\cdot)\|_{\mathcal{U}}^2 = \int_0^T u^{\mathsf{T}}(t) u(t)\, \mathrm{d}t.$$

With every control function $u(t) \in \mathcal{U}$ we associate the corresponding state trajectory $q(t) = \varphi_{q_0,t}(u(\cdot))$ and

the task space trajectory $y(t) = k(\varphi_{q_0,t}(u(\cdot)))$, where $\varphi_{q_0,t}(u(\cdot))$ denotes the flow of the system (1) at the moment t, initialized in q_0 and driven by the control function $u(\cdot)$.

The endogenous configuration space approach establishes an analogy between the holonomic redundant manipulators and the nonholonomic robots. Following this line of reasoning, we can define the kinematics as the end point map $K_{q_0,T}\colon \mathcal{U} \to \mathbf{R}^r$ of the system (1) as

$$K_{q_0,T}\big(u(\cdot)\big) = y(T) = k\Big(\varphi_{q_0,T}\big(u(\cdot)\big)\Big). \tag{3}$$

This map computes the final point $y(T)$ of the output of (1) under the influence of the control $u(\cdot)$.

Still following the analogy to the manipulation robots, we can differentiate the end point map to obtain the Jacobian $J_{q_0,T}\colon \mathcal{U} \to \mathbf{R}^r$ [19]

$$\begin{aligned} J_{q_0,T}\big(u(\cdot)\big)v(\cdot) &= \mathcal{D}K_{q_0,T}\big(u(\cdot)\big)v(\cdot) \\ &= C(T)\int_0^T \Phi(T,s)B(s)v(s)\,\mathrm{d}s, \end{aligned} \tag{4}$$

where the matrices $A(t) = \frac{\partial(f(q)+G(q)u)}{\partial q}$, $B(t) = G(q)$, $C(t) = \frac{\partial k(q)}{\partial q}$ come from the associated linear system, which is actually the linear approximation to (1) along a control–trajectory $\big(u(\cdot), q(\cdot)\big)$ pair

$$\begin{cases} \frac{\mathrm{d}\xi(t)}{\mathrm{d}t} = A(t)\xi + B(t)v(t), \\ \zeta(t) = C(t)\xi, \end{cases} \qquad \xi(0) = 0, \tag{5}$$

and $\Phi(t,s)$ is the fundamental matrix of (5) solving the partial differential equation $\frac{\partial\Phi(t,s)}{\partial t} = A(t)\Phi(t,s)$ with the boundary condition $\Phi(s,s) = I_n$ [17]. The Jacobian (4) determines how the output $y(T)$ will change in response to infinitesimal changes $v(\cdot)$ of the control function $u(\cdot)$.

Before we define the Jacobian motion planning algorithm, we have to introduce the Jacobian inverse and the adjoint Jacobian. Let us begin with the Jacobian inverse. As we have already mentioned, the Jacobian (4) transforms the variations $v(\cdot) \in \mathcal{U}$ of the endogenous configuration into variations $\eta \in \mathbf{R}^r$ in the task space, so we can write the Jacobian equation

$$J_{q_0,T}\big(u(\cdot)\big)v(\cdot) = C(T)\int_0^T \Phi(T,s)B(s)v(s)\,\mathrm{d}s = \eta. \tag{6}$$

As long as we stay within the region of regular configurations, i.e., the $J_{q_0,T}\big(u(\cdot)\big)$ is surjective, we can solve (6) with respect to $v(\cdot)$ and obtain a Jacobian inverse. Using the least squares method and the Lagrange multiplier technique, one can derive the Jacobian pseudoinverse (Moore–Penrose) $J^{\#}_{q_0,T}\big(u(\cdot)\big)\colon \mathbf{R}^r \to \mathcal{U}$ as [19]

$$\Big(J^{\#}_{q_0,T}\big(u(\cdot)\big)\eta\Big)(t) = B^{\mathsf{T}}(t)\Phi^{\mathsf{T}}(T,t)C^{\mathsf{T}}(T)\mathcal{G}^{-1}_{q_0,T}\big(u(\cdot)\big)\eta, \tag{7}$$

where

$$\begin{aligned} &\mathcal{G}_{q_0,T}\big(u(\cdot)\big) \\ &\quad = C(T)\int_0^T \Phi(T,s)B(s)B^{\mathsf{T}}(s)\Phi^{\mathsf{T}}(T,s)\,\mathrm{d}s\,C^{\mathsf{T}}(T) \end{aligned} \tag{8}$$

is the Gram matrix which can be used to distinguish regular and singular configurations. If the Gram matrix has full rank then the configuration is regular and the system (1) is locally (along the control–trajectory pair) controllable. The Gram matrix could be computed by solving the Lyapunov differential equation

$$\frac{\mathrm{d}M(t)}{\mathrm{d}t} = B(t)B^{\mathsf{T}}(t) + A(t)M(t) + M(t)A^{\mathsf{T}}(t) \tag{9}$$

with the initial condition $M(0) = 0$, and then substituting $\mathcal{G}_{q_0,T}(u(\cdot)) = C(T)M(T)C^{\mathsf{T}}(T)$.

Finally, we introduce the adjoint Jacobian $J^{*}_{q_0,T}(u(\cdot))\colon (\mathbf{R}^r)^* \to (\mathcal{U})^*$, which transforms the dual space to the task space into the dual configuration space in accordance with the formula

$$\Big\langle J^{*}_{q_0,T}\big(u(\cdot)\big)\eta,\, v(\cdot)\Big\rangle = \eta^{\mathsf{T}} J_{q_0,T}\big(u(\cdot)\big)v(\cdot), \tag{10}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product (2). Using the formula (10) we derive the adjoint Jacobian [19]

$$\Big(J^{*}_{q_0,T}\big(u(\cdot)\big)\eta\Big)(t) = B^{\mathsf{T}}(t)\Phi^{\mathsf{T}}(T,t)C^{\mathsf{T}}(T)\eta. \tag{11}$$

Invoking the definition of the adjoint Jacobian (11), the Gram matrix (8) can be rewritten as

$$\mathcal{G}_{q_0,T}\big(u(\cdot)\big) = J_{q_0,T}\big(u(\cdot)\big)J^{*}_{q_0,T}\big(u(\cdot)\big), \tag{12}$$

and the Jacobian pseudoinverse (7) as

$$J^{\#}_{q_0,T}\big(u(\cdot)\big) = J^{*}_{q_0,T}\big(u(\cdot)\big)\mathcal{G}^{-1}_{q_0,T}\big(u(\cdot)\big). \tag{13}$$

Observe that (13) resembles the classic formula $J^{\#} = J^{\mathsf{T}}(JJ^{\mathsf{T}})^{-1}$.

In the redundant manipulators, the number of additional tasks is limited by the degree of redundancy (the difference between the number of joints and the dimension of the task space). In the endogenous configuration space approach, the Jacobian (4) operates on the infinite-dimensional endogenous configuration space. This means that in the case of the nonholonomic systems the so-called "degree of redundancy" is infinite and the number of additional subtasks could be theoretically unlimited.

3 Jacobian motion planning algorithm

In this section, we shall introduce the motion planning algorithm based on the previously defined Jacobian. Firstly, we provide a derivation of the proper motion planning algorithm. Then, we expand the approach and define two multiple task algorithms which will plan the motion simultaneously with one or more additional tasks.

Let us introduce some nomenclature. The proper motion planning algorithm solves the proper motion planning problem which consists only of the motion planning task. The multiple task motion planning algorithm can solve the proper motion planning problem as well as one or more additional tasks. These additional tasks will be called subtasks. For the clarity of presentation, we assume that the subtask with index 0 is the proper motion planning task, and the total number of subtasks is denoted with $s+1$. Each subtask will be defined by its task map ${}^{i}K_{q_0,T}(u(\cdot))$, $i = 0, 1, \ldots, s$. For the proper motion planning ($i = 0$), the task map is defined by (3). The other task maps as well as the corresponding Jacobians, their inverses, and the subtask errors will be defined later on.

The definition of the proper motion planning problem for system (1) is as follows: Find a control function $u(\cdot) \in \mathcal{U}$ which drives the system (1) from a certain initial space configuration $q(0)$ to desired point $y_d \in \mathbf{R}^r$ in the task space in a prescribed time interval $[0, T]$. With such defined problem, we can associate the error formula as $e(u(\cdot)) = K_{q_0,T}(u(\cdot)) - y_d = {}^{0}K_{q_0,T}(u(\cdot)) - y_d$, where $K_{q_0,T}(u(\cdot))$ is defined in (3). We want to solve such a problem using the continuation method [18]. For this reason, we choose in the control space $\mathcal{U}$ a smooth curve $u_\vartheta(\cdot) \in \mathcal{U}$ parametrized by $\vartheta \in \mathbf{R}$, passing through a certain initial configuration (control) $u_0(\cdot)$. For that curve, we define the motion planning error as

$$e(\vartheta) = K_{q_0,T}\big(u_\vartheta(\cdot)\big) - y_d. \tag{14}$$

Let us assume that the error should decrease exponentially along the curve with a decay rate $\gamma > 0$

$$\frac{\mathrm{d}e(\vartheta)}{\mathrm{d}\vartheta} = -\gamma e(\vartheta). \tag{15}$$

The substitution of the (14) into (15) yields the Ważewski–Davidenko equation

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}\vartheta}\big(K_{q_0,T}\big(u_\vartheta(\cdot)\big) - y_d\big)\\ &\quad = J_{q_0,T}\big(u_\vartheta(\cdot)\big)\frac{\mathrm{d}}{\mathrm{d}\vartheta}u_\vartheta(\cdot) = -\gamma e(\vartheta).\end{aligned} \tag{16}$$

To solve equation (16) we can use any right Jacobian inverse. If we use the Jacobian pseudoinverse (7) then the dynamic system of the proper motion planning algorithm can be expressed as

$$\frac{\mathrm{d}u_\vartheta(t)}{\mathrm{d}\vartheta} = -\gamma\Big(J^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big)e(\vartheta)\Big)(t). \tag{17}$$

The solution of the motion planning problem is the control function obtained as a limit $\lim_{\vartheta\to\infty} u_\vartheta(\cdot)$ of the resultant trajectory of (17). In the following subsections, we shall present two ways of modifying the proper motion planning algorithm in order to solve the motion planning problem with additional tasks.

3.1 Multiple task motion planning

The multiple task motion planning algorithms can be twofold. An egalitarian algorithm treats all subtasks equivalently, while a prioritarian algorithm arranges the subtasks with decreasing priorities. The next two subsections introduce these two types of algorithms.

3.1.1 Egalitarian algorithm

As was already mentioned, the egalitarian approach to motion planning treats the proper motion planning task

as well as all the additional subtasks as equally important. To derive the egalitarian algorithm, let us begin with the definition of the collective error

$$\mathbf{e}(\vartheta) = \big({}^0e(\vartheta), {}^1e(\vartheta), \ldots, {}^se(\vartheta)\big), \tag{18}$$

where ${}^0e(\vartheta)$ is the proper motion planning task error (14) and ${}^ie(\vartheta) = {}^iK_{q_0,T}(u_\vartheta(\cdot)) \in \mathbf{R}^{r_i}$ for $i = 1, 2, \ldots, s$ stand for the errors of the subtasks. Assuming that the error should decrease exponentially with the decay rate γ, we can write the Ważewski–Davidenko equation

$$\mathbf{J}_{q_0,T}\big(u_\vartheta(\cdot)\big)\frac{\mathrm{d}u_\vartheta(\cdot)}{\mathrm{d}\vartheta} = -\gamma\mathbf{e}(\vartheta), \tag{19}$$

where

$$\mathbf{J}_{q_0,T}\big(u(\cdot)\big) = \begin{bmatrix} {}^0J_{q_0,T}\big(u(\cdot)\big) \\ {}^1J_{q_0,T}\big(u(\cdot)\big) \\ \vdots \\ {}^sJ_{q_0,T}\big(u(\cdot)\big) \end{bmatrix} \tag{20}$$

is a collective Jacobian, ${}^0J_{q_0,T}(u(\cdot))$ is defined by (4), and ${}^iJ_{q_0,T}(u(\cdot))v(\cdot) = \mathcal{D}\,{}^iK_{q_0,T}(u(\cdot))v(\cdot)$, $i = 1, 2, \ldots, s$ is the Jacobian for the i-th subtask. Using the pseudoinverse of the collective Jacobian (20)

$$\mathbf{J}^{\#}_{q_0,T}\big(u(\cdot)\big) = \mathbf{J}^{*}_{q_0,T}\big(u(\cdot)\big)\Big(\mathbf{J}_{q_0,T}\big(u(\cdot)\big)\mathbf{J}^{*}_{q_0,T}\big(u(\cdot)\big)\Big)^{-1}, \tag{21}$$

where the collective adjoint Jacobian $\mathbf{J}^{*}_{q_0,T}(u(\cdot))$ is defined analogously to (10), one can solve equation (19) and obtain the egalitarian motion planning algorithm as a dynamic system

$$\frac{\mathrm{d}u_\vartheta(t)}{\mathrm{d}\vartheta} = -\gamma\Big(\mathbf{J}^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big)Ee(\vartheta)\Big)(t), \tag{22}$$

where $E = \mathrm{blockdiag}\{\epsilon_i I_{r_i}\}$ is a block diagonal weight matrix, which allows us to scale the influence exerted by the particular subtask on the problem solution, by tunning the ϵ_i value. The control function which solves the egalitarian motion planning problem is computed as a limit $\lim_{\vartheta\to\infty} u_\vartheta(\cdot)$ of the solution of the system (22).

3.1.2 Prioritarian algorithm

Differently to the egalitarian approach, the prioritarian approach assigns to each subtask a priority with decreasing order. The subtask with a higher priority is taken into consideration before the subtasks with lower priorities. It also means that in case when the solution of lower priority subtask would influence the solution of the higher priority task, the lower priority task may not be solved. Let us assume that the subtask index indicates also the priority (0 means the highest priority). For the i-th subtask, we can write

$$\frac{\mathrm{d}\,{}^ie(\vartheta)}{\mathrm{d}\vartheta} = {}^iJ_{q_0,T}\big(u_\vartheta(\cdot)\big)\frac{\mathrm{d}u_\vartheta(\cdot)}{\mathrm{d}\vartheta}. \tag{23}$$

Assuming that the error should decrease exponentially, one can write the algorithm which solves (23) using the Jacobian generalized inverse [3,19]

$$\begin{aligned}\frac{\mathrm{d}u_\vartheta(t)}{\mathrm{d}\vartheta} = &-{}^i\gamma\Big({}^iJ^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^ie(\vartheta)\Big)(t) \\ &+\Big({}^iP_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^i\mu_\vartheta(\cdot)\Big)(t),\end{aligned} \tag{24}$$

where

$${}^iP_{q_0,T}\big(u_\vartheta(\cdot)\big) = \mathrm{id}_{\mathcal{U}} - {}^iJ^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big){}^iJ_{q_0,T}\big(u_\vartheta(\cdot)\big) \tag{25}$$

is a projection of $\mathcal{U}$ onto $\ker {}^iJ_{q_0,T}(u_\vartheta(\cdot))$, ${}^i\mu_\vartheta(\cdot) \in \mathcal{U}$ and $\mathrm{id}_{\mathcal{U}}$ is the identity map. Following [5], let us reformulate the derivation of the prioritarian multiple task motion planning algorithm for nonholonomic systems. For any two subtasks, say subtasks 1 and 2, the control function $u_\vartheta(\cdot)$ has to be the same, so we can write from (24)

$$\begin{aligned}&\Big(-{}^1\gamma\,{}^1J^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^1e(\vartheta) + {}^1P_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^1\mu_\vartheta(\cdot)\Big)(t) \\ &= \Big(-{}^2\gamma\,{}^2J^{\#}_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^2e(\vartheta) + {}^2P_{q_0,T}\big(u_\vartheta(\cdot)\big)\,{}^2\mu_\vartheta(\cdot)\Big)(t).\end{aligned} \tag{26}$$

Using the projection property of: idempotence ${}^iP_{q_0,T}(u_\vartheta(\cdot)){}^iP_{q_0,T}(u_\vartheta(\cdot)) = {}^iP_{q_0,T}(u_\vartheta(\cdot))$, symmetry ${}^iP_{q_0,T}(u_\vartheta(\cdot)) = {}^iP^{\mathsf{T}}_{q_0,T}(u_\vartheta(\cdot))$, and the annihilation ${}^iP_{q_0,T}(u_\vartheta(\cdot)){}^iJ^{\#}_{q_0,T}(u_\vartheta(\cdot)) = 0$, the (26) could be rearranged as follows

$$\begin{aligned}&\left({}^1P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^1\mu_\vartheta(\cdot)\right)(t)\\&\quad=-{}^2\gamma\left({}^1P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2J^{\#}_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2e(\vartheta)\right)(t)\\&\qquad+\left({}^1P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2\mu_\vartheta(\cdot)\right)(t).\end{aligned}\tag{27}$$

Substituting (27) into (24) with $i=1$ we arrive at the algorithm

$$\begin{aligned}\frac{\mathrm{d}u_\vartheta(t)}{\mathrm{d}\vartheta}&=-{}^1\gamma\left({}^1J^{\#}_{q_0,T}\left(u_\vartheta(\cdot)\right){}^1e(\vartheta)\right)(t)\\&\quad-{}^2\gamma\left({}^1P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2J^{\#}_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2e(\vartheta)\right)(t)\\&\quad+\left({}^1P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2P_{q_0,T}\left(u_\vartheta(\cdot)\right){}^2\mu_\vartheta(\cdot)\right)(t),\end{aligned}\tag{28}$$

which, if we set ${}^2\mu(\cdot)=0$, is the prioritarian algorithm for two subtasks. The endogenous configuration ${}^2\mu(\cdot)$ could be used to expand the algorithm for more subtasks. Finally, the prioritarian algorithm for s-subtasks takes the form of the dynamic system [13]

$$\begin{aligned}\frac{\mathrm{d}u_\vartheta(t)}{\mathrm{d}\vartheta}&=-\left(\sum_{i=0}^{s}{}^i\gamma\left(\prod_{j=0}^{i}{}^{j-1}P_{q_0,T}\left(u_\vartheta(\cdot)\right)\right)\right.\\&\quad\left.\times{}^iJ^{\#}_{q_0,T}\left(u_\vartheta(\cdot)\right){}^ie(\vartheta)\right)(t),\end{aligned}\tag{29}$$

where ${}^{-1}P_{q_0,T}\left(u_\vartheta(\cdot)\right)=\mathrm{id}_{\mathcal{U}}$. In terms of [1] this algorithm is a successive inverse-based projection method. Similar to the previous algorithms, the solution of the motion planning problem is a control function obtained as a limit $\lim_{\vartheta\to\infty}u_\vartheta(\cdot)$ of the trajectory of (29). In the prioritarian algorithm, the significance of a particular subtask is controlled by the corresponding decay rate ${}^i\gamma$.

3.2 Comparison remarks

Having defined the two multiple task motion planning algorithms, namely the egalitarian (22) and the prioritarian (29), we can provide some comparison remarks. The analysis of the algorithm formulas shows that in the egalitarian approach we have to compute the inverse operator $\mathbf{J}^{\#}_{q_0,T}\colon\mathbf{R}^{\bar r}\to\mathcal{U}$ where $\bar r=\sum_{i=0}^{s}r_i$. On the other hand, in the prioritarian algorithm the multiple inverse operators of the form ${}^iJ^{\#}_{q_0,T}\colon\mathbf{R}^{r_i}\to\mathcal{U}$ must be found. Taking these two facts into account, one can observe that the prioritarian approach is a kind of decomposition of the larger problem while the egalitarian approach tries to solve the entire problem at once. However, the prioritarian algorithm is usually more difficult to implement, for example due to the projection $P_{q_0,T}$ calculation.

The construction of the two multiple motion planning algorithms imposes the usage for specific problems. If the subtasks can be arranged with decreasing priorities, then the prioritarian approach should be preferred. As it was already written, the prioritarian algorithm may return the solution which does not solve all of the subtasks. The egalitarian algorithm works differently, it tries to solve all subtasks simultaneously. However, when the solution of all subtasks does not exists at all, or the subtasks compete with each other, then the egalitarian algorithm will fail. In such case, the prioritarian approach will return the best possible solution. As long as the subtasks are equivalent in the egalitarian algorithm, it is possible that one of the subtasks (let us say less important) will dominate the solution. It is even possible to obtain some oscillations, when the subtasks start to compete. When the error of one subtask dominates the other one, then the algorithm tries to reduce that error. When it becomes smaller, then the other one starts to dominate. Such situation leads to no solution. When the prioritarian algorithm is applied the oscillations are not possible.

4 Subtasks

As was already written, the subtasks are described by the task maps ${}^iK_{q_0,T}(u(\cdot))\colon\mathcal{U}\to\mathbf{R}^{r_i}$. For the construction of the multiple task motion planning algorithm, it will also be necessary to define the subtask Jacobian ${}^iJ_{q_0,T}(u(\cdot))$, its inverse $({}^iJ^{\#}_{q_0,T}(u(\cdot)))(t)$, and the subtask's error ${}^ie(\vartheta)$.

Let us assume that the i-th task map have the following form

$${}^iK_{q_0,T}\left(u(\cdot)\right)=\int_0^T{}^i\alpha\left(q(t),y(t),u(t)\right)\mathrm{d}t,\tag{30}$$

where ${}^i\alpha(q(t),y(t),u(t))\geq 0$ is smooth and will be denoted for short as ${}^i\alpha(t)$. For such a task map the corresponding Jacobian takes the form [14]

$${}^iJ_{q_0,T}\left(u(\cdot)\right)v(\cdot)=\int_0^T\left(\frac{\partial\,{}^i\alpha(t)}{\partial q}\xi(t)+\frac{\partial\,{}^i\alpha(t)}{\partial u}v(t)\right)\mathrm{d}t,\tag{31}$$

and its inverse computed in [14] is

$$\left({}^{i}J^{\#}_{q_0,T}\big(u(\cdot)\big)v(\cdot)\eta\right)(t) = {}^{i}\beta(t)\left(\left\|{}^{i}\beta(\cdot)\right\|^{2}_{\mathcal{U}}\right)^{-1}\eta, \quad (32)$$

where ${}^{i}\beta(t)$ is defined as follows

$${}^{i}\beta(t) = B^{\mathsf{T}}(t)\int_{t}^{T}\Phi^{\mathsf{T}}(s,t)\left(\frac{\partial\,{}^{i}\alpha(s)}{\partial q}\right)^{\mathsf{T}}\mathrm{d}s + \left(\frac{\partial\,{}^{i}\alpha(t)}{\partial u}\right)^{\mathsf{T}}. \quad (33)$$

The function ${}^{i}\beta(t)$ may be computed by solving the following differential equation

$$\frac{\mathrm{d}\,{}^{i}b(t)}{\mathrm{d}t} = -A(t)^{\mathsf{T}}\,{}^{i}b(t) - \left(\frac{\partial\,{}^{i}\alpha(t)}{\partial q}\right)^{\mathsf{T}}, \quad (34)$$

with the final condition ${}^{i}b(T) = 0$, and then substituting the solution of (34) into

$${}^{i}\beta(t) = B^{\mathsf{T}}(t)\,{}^{i}b(t) + \left(\frac{\partial\,{}^{i}\alpha(t)}{\partial u}\right)^{\mathsf{T}}. \quad (35)$$

For every subtask with $i = 1, 2, \ldots, s$ we assume that the subtask error is determined by the subtask end point map, i.e., ${}^{i}e(\vartheta) = {}^{i}K_{q_0,T}\big(u_{\vartheta}(\cdot)\big)$. According to the above formulas, to define any subtask it is necessary to define the functions ${}^{i}\alpha(t)$ and ${}^{i}\beta(t)$. In this paper we shall introduce three different subtasks.

1. Control energy minimization—is a subtask which minimizes the total control energy.
2. State variable minimization—in this subtask the value of one or more state variables should be as close to 0 as possible during the motion time.
3. Obstacle/Singularity avoidance—this subtask introduces an obstacle function $h(y)$ which describes the locations and shapes of the obstacles. If the obstacles are defined in the task space, we have "classical" obstacle avoidance problem. If the obstacles are defined in the state space then the problem could be treated as a kind of the singularity avoidance problem.

Table 1 collects all necessary functions to completely define these three types of subtasks, that could be treated as basic.

The first one represents the function which depends on control signals $u(t)$, the second one depends on the state trajectory $q(t)$, and the third function depends on the output trajectory $y(t)$. The matrix $\sigma(s)$ is a diagonal weight matrix which allows us to control the influence on the solution of the particular components of the vector $u(t)$ or $q(t)$. Obviously, one can define any other subtask by combining the basic ones. The example of control energy minimization is presented in [13]. The singularity avoidance could be found in [14]. The work [21] introduces a subtask corresponding to a reduction of the wheels slip.

Table 1 Functions ${}^{i}\alpha(t)$ and ${}^{i}\beta(t)$ for various subtasks

Task	${}^{i}\alpha(t)$	${}^{i}\beta(t)$
1	$u^{\mathsf{T}}(t)\sigma(t)u(t)$	$\sigma(t)u(t)$
2	$q^{\mathsf{T}}(t)\sigma(t)q(t)$	$B^{\mathsf{T}}(t)\int_{t}^{T}\Phi^{\mathsf{T}}(s,t)\sigma(s)q(s)\,\mathrm{d}s$
3	$h(y(t))$	$B^{\mathsf{T}}(t)\int_{t}^{T}\Phi^{\mathsf{T}}(s,t)C^{\mathsf{T}}(s)\left(\frac{\partial h(y(s))}{\partial y}\right)^{\mathsf{T}}\mathrm{d}s$

5 Numerical example

Now, we shall shift our attention toward the numerical aspects of the multiple task motion planning algorithms. All computations were accomplished in the MATLAB environment. We shall describe our computation methods and make some comments on the organization of computation. Later on, we shall present some simulation results and assess the algorithms efficiency.

5.1 Computational aspects

The algorithm equation (22) as well as (29) is a specific functional differential equation. To solve this kind of equation a specific computation methodology needs to be adopted. Till now [9,13,19,20], the solution of motion planning algorithm with endogenous configuration space approach has been mostly obtained by control function parametrization (truncated series parametrization) and by the algorithm discretization (fixed-step-size Euler method). The recent research [15] shows that the computations could be made in a more effective way. In this work, the multiple task algorithms are implemented in the nonparametric version, assisted with higher order algorithms of differential equations integration. As mentioned before, the

equation underlying the motion planning algorithms is a functional differential equation depending on two variables t and ϑ. To provide the necessary computations, we proceed along the following lines:

1. For the initial value $\vartheta = 0$ choose an arbitrary initial control function $u_{\vartheta=0}(t) = u_0(t) \in \mathcal{U}$.
2. Apply the control $u_\vartheta(t)$ to control affine system (1) to obtain the error values.
3. If the motion planning error ${}^0e(\vartheta)$ (14) and the errors of the other subtasks ${}^ie(\vartheta)$ are below the assumed limits then the problem is solved, otherwise proceed to 4.
4. Compute a new control function $u_\vartheta(t)$ from the algorithm equation (22) or (29) for the next value of ϑ and return to 2.

Proceedings as above, especially with respect to item 4, for the computation of the control function $u_\vartheta(t)$ as long as it depends on ϑ, it is necessary to solve for every ϑ the following differential–algebraic system of equations

$$\begin{cases} \frac{\mathrm{d}q_\vartheta(t)}{\mathrm{d}t} = f(q_\vartheta(t)) + G(q_\vartheta(t))u_\vartheta(t), \\ \frac{\partial \Phi_\vartheta(T,t)}{\partial t} = -\Phi_\vartheta(T,t)A_\vartheta(t), \\ \frac{\mathrm{d}M_\vartheta(t)}{\mathrm{d}t} = B_\vartheta(t)B_\vartheta^{\mathsf{T}}(t) + A_\vartheta(t)M_\vartheta(t) + M_\vartheta(t)A_\vartheta^{\mathsf{T}}(t), \\ \mathcal{G}_{q_0,T}(u(\cdot)) = C_\vartheta(T)M_\vartheta(T)C_\vartheta^{\mathsf{T}}(T), \\ A_\vartheta(t) = \frac{\partial(f(q_\vartheta(t))+G(q_\vartheta(t))u_\vartheta(t))}{\partial q}, \\ B_\vartheta(t) = G(q_\vartheta(t)),\ C_\vartheta(t) = \frac{\partial k(q_\vartheta(t))}{\partial q}, \\ {}^0e(\vartheta) = K_{q_0,T}(u_\vartheta(\cdot)) - y_d,\quad {}^ie(\vartheta) = {}^iK_{q_0,T}(u_\vartheta(\cdot)), \\ \frac{\mathrm{d}b_\vartheta(t)}{\mathrm{d}t} = -A_\vartheta(t)b_\vartheta(t) - \left(\frac{\partial\, {}^i\alpha_\vartheta\left(q_\vartheta(t),y_\vartheta(t),u_\vartheta(t)\right)}{\partial q}\right)^{\mathsf{T}}, \\ {}^i\beta_\vartheta(t) = B_\vartheta^{\mathsf{T}}(t)b_\vartheta(t) + \left(\frac{\partial\, {}^i\alpha_\vartheta\left(q_\vartheta(t),y_\vartheta(t),u_\vartheta(t)\right)}{\partial u}\right)^{\mathsf{T}}, \end{cases} \tag{36}$$

with initial conditions $q_\vartheta(0) = q_0$, $M_\vartheta(0) = 0$, the final condition $b_\vartheta(T) = 0$ and the boundary condition $\Phi(T,T) = I_n$. The resultant trajectories from (36) could be used to write every ingredient of the egalitarian multiple task motion planning algorithm (22) or the prioritarian multiple task motion planning algorithm (29). To summarize, the computations are organized in the form of two nested differential equation solvers. The inner solver computes the solution of the system (36) for $t \in [0,T]$, while the outer solver determines the resultant trajectory of the algorithm equation (22) or (29) for $\vartheta \to +\infty$. An outline of the pseudocode illustrating the above procedure may look as follows

% initialization
READ $\theta_{\max}, u_0(t), \Delta_{\vartheta,\text{init}}, \Delta_{t,\text{init}}, T, q_0, e_{\max}$
$\Delta_\vartheta = \Delta_{\vartheta,\text{init}}$
$\vartheta = 0$
$u_\theta(t) = u_0(t)$
REPEAT *% outer solver*
  % initialization
  $\Delta_t = \Delta_{t,\text{init}}, t = 0$
  $q_\vartheta(0) = q_0, \Phi_\vartheta(T,T) = I_n$
  $M_\vartheta(0) = 0, b_\vartheta(T) = 0$
  REPEAT *% inner solver*
    Compute RHS of (36) for current t
    Δ_t=OPTIMAL[t] *% Correct if necessary*
    $t = t + \Delta_t$
  UNTIL $t \geq T$
  % updating $u_\vartheta(t)$
  Compute RHS of (22) or (29) for current ϑ
  Δ_θ =OPTIMAL[ϑ] *% Correct if necessary*
  $\vartheta = \vartheta + \Delta_\vartheta$
UNTIL $(\theta \geq \theta_{\max})$ or $(\|e(\vartheta)\| < e_{\max})$

These two solvers are built-in MATLAB variable step methods (e.g., `ode45`), thus the computation accuracy could be easily controlled. In the pseudocode this fact is denoted with the function OPTIMAL, which returns the optimal step length to achieve the demanded accuracy. In theory, the solution of the multiple task motion planning algorithm is obtained as a limit $\lim_{\vartheta\to\infty} u_\vartheta(\cdot)$, practically we stop the computation when the errors drop below the assumed levels $(e_{\max})$.

5.2 Simulation results

As a testbed we have chosen a simplified model of the underacutated surface vessel, often called unmanned surface vessel (USV) [10] which is a subclass of autonomous underwater vehicle (AUV) [7]. Such a system could be regarded as a hovercraft model.

If we assume that the body of the vessel has the disk shape, and the propellers are located at the center of mass (Fig. 1) then the dynamic equation of motion takes the form [8,12]

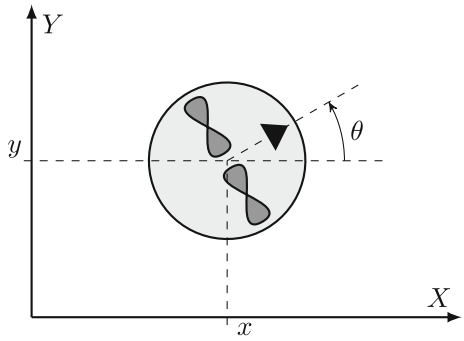

Fig. 1 Unmanned surface vessel

$$\begin{cases} \dot{x} = \nu_u \cos\theta - \nu_v \sin\theta, \\ \dot{y} = \nu_u \sin\theta + \nu_v \cos\theta, \\ \dot{\theta} = \nu_r, \\ \dot{\nu}_u = \nu_v \nu_r + u_u, \\ \dot{\nu}_v = -\nu_v \nu_r, \\ \dot{\nu}_r = u_r, \end{cases} \tag{37}$$

where the $q = (x, y, \theta, \nu_u, \nu_v, \nu_r)$ is the state space vector. The x, y and θ represent the position and orientation, ν_u, ν_v and ν_r denote the linear surge velocity, linear sway velocity and angular yaw velocity, respectively. We assume that only the control force in surge u_u and control torque in yaw u_r are available. The system (37) can be transformed into the affine control system (1) with the following vector fields

$$f(q) = \begin{pmatrix} q_4 \cos q_3 - q_5 \sin q_3 \\ q_4 \sin q_3 + q_5 \cos q_3 \\ q_6 \\ q_5 q_6 \\ -q_4 q_6 \\ 0 \end{pmatrix}, \qquad G(q) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{38}$$

To assess efficiency of the two multiple task motion planning algorithms, we create three simulation scenarios. In each scenario, a different type of subtasks is considered. In every case, the problem is solved using first the egalitarian and then the prioritarian algorithm. For the comparison, we also add a solution of the motion planning problem obtained by the single task (only proper motion planning) algorithm. In all figures in the following subsection, we use the corresponding indexes to distinguish the algorithms. So, the single task algorithm is denoted with "S", the prioritarian algorithm is marked with "P" and the index "E" reffers to the egalitarian algorithm.

5.2.1 Motion planning with control energy minimization

In the first problem we want to reach a destination point together with the control energy minimization as a minimization of the integral $\int_0^T u^{\mathsf{T}}(t)\sigma(t)u(t)\,\mathrm{d}t$. The simulation parameters are collected in Table 2.

The selected form of matrix $\sigma(t)$ means that we are interested only in reducing the energy of the second control $u_2(t)$, namely the yaw torque. The simulation results are depicted in Figs. 2, 3, 4 and 5. The motion path is presented in Fig. 2. It can be observed that both multiple task algorithms return a wider motion path than the path from the single task motion planning algorithm. The result of the control energy minimization could also be seen in Fig. 3 where the plots of the control function $u_2(t)$ are shown. It can be observed, that using the egalitarian approach the amplitude of control function is smaller than in the prioritarian case. Figures. 4 and 5 present the convergence of the proper motion planning task error and of the control minimization task error, respectively. Here also, one can see that the egalitarian approach produces smaller error of the control minimization task ${}^1e(\vartheta)$ (see Fig. 5). On the

Table 2 Motion planning with control energy minimization—simulation parameters

q_0	q_d	T	$({}^0\gamma, {}^1\gamma)$	E	$\sigma(t)$	$u_0(t)$
$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 5 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	5	(1, 10)	$\begin{bmatrix} I_6 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{pmatrix} \exp(-t) \\ \exp(-t) \end{pmatrix}$

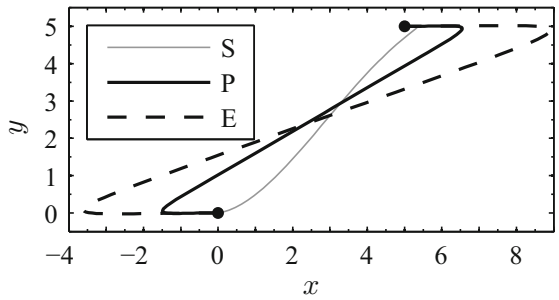

Fig. 2 Motion planning with control energy minimization—path in XY plane

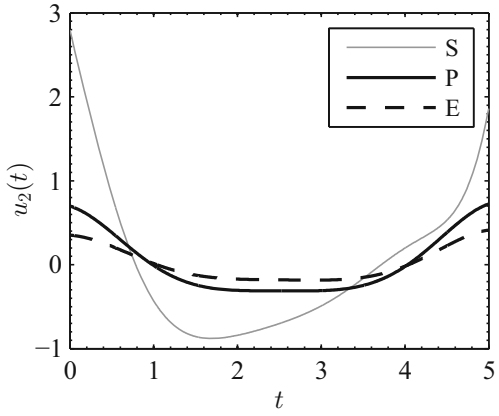

Fig. 3 Motion planning with control energy minimization—control $u_2(t)$

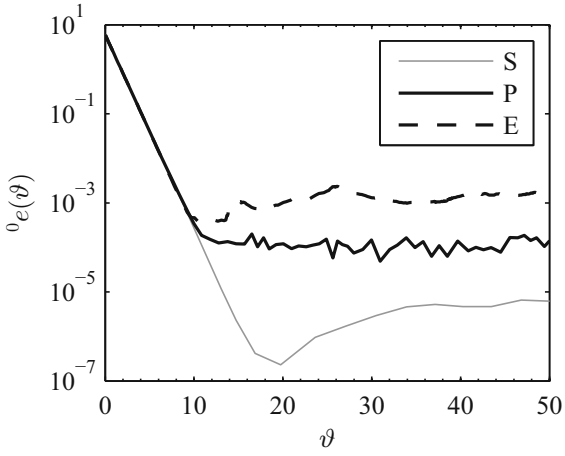

Fig. 4 Motion planning with control energy minimization—convergence of ${}^0e(\vartheta)$

other hand, the proper motion planning task is solved more accurately by the prioritarian algorithm (the error ${}^0e(\vartheta)$ in Fig. 4 takes smaller values for the prioritarian approach than the egalitarian). Figures 4 and 5 present also the main difference between these two approaches. The prioritarian algorithm allows the additional subtask error to temporary increase as long as the first task error convergence is not interrupted. Differently, in the egalitarian approach both errors have to decrease simultaneously and the influence of the ${}^1e(\vartheta)$ on the ${}^0e(\vartheta)$ results in higher value of ${}^0e(\vartheta)$ than for the prioritarian approach (see Fig. 4). The saturation of the ${}^0e(\vartheta)$ error decrease, which can be seen in Fig. 4, depends on the computation accuracy. The improvement of the computation accuracy results in decreasing the saturation

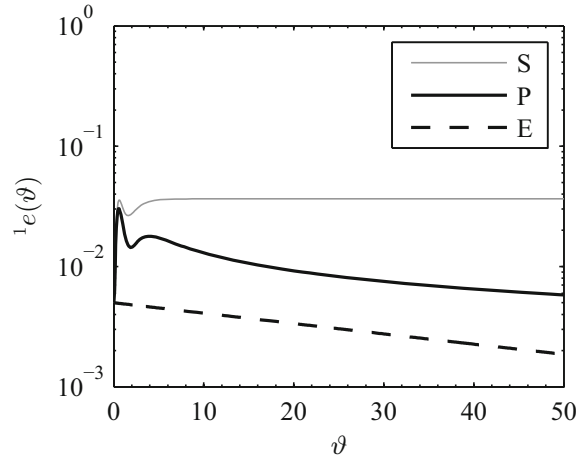

Fig. 5 Motion planning with control energy minimization—convergence of ${}^1e(\vartheta)$

level, but on the other hand, it increases the computation time.

5.2.2 *Motion planning with state variable value minimization*

The second problem consists in finding a control function which drives the model from an initial to a desired point, together with the minimization of a state variable in the meaning of the minimization of the integral $\int_0^T q^{\mathsf{T}}(t)\sigma(t)q(t)\,\mathrm{d}t$. Table 3 collects all parameters used in the simulation. We want to minimize only the state variable $q_5(t) = \nu_v(t)$, by choosing a suitable $\sigma(t)$ matrix. It means that in the whole motion the linear sway velocity should be as small as possible. Figures from 6 to 9 present the results of the simulation. The path in XY plane (Fig. 6) shows the contribution of the additional subtask to the motion planning problem solution. The resultant paths obtained from both algorithms, egalitarian and prioritarian, are almost the straight lines. At the beginning of the motion, the vessel changes its orientation, next it goes ahead toward the desired point, and finally the orientation is changed again, so the linear sway velocity is close to zero during the whole motion time, as can be observed in Fig. 7. For comparison, the solution of the single task algorithm is also depicted. It is obvious that both the egalitarian and the prioritarian multiple task algorithms outperform the single task algorithm. Figure 8 displays the convergence of the main task error ${}^0e(\vartheta)$, and the additional subtask error ${}^1e(\vartheta)$ is plotted in Fig. 9. The value

Table 3 Motion planning with state variable value minimization—simulation parameters

q_0	q_d	T	$({}^0\gamma, {}^1\gamma)$	E	$\sigma(t)$	$u_0(t)$
$\begin{pmatrix}0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}2\\2\\\pi\\0\\0\\0\end{pmatrix}$	5	(10, 10)	I_7	diag $\begin{pmatrix}0\\0\\0\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0.3\\0.1\sin(2\pi t/T)\end{pmatrix}$

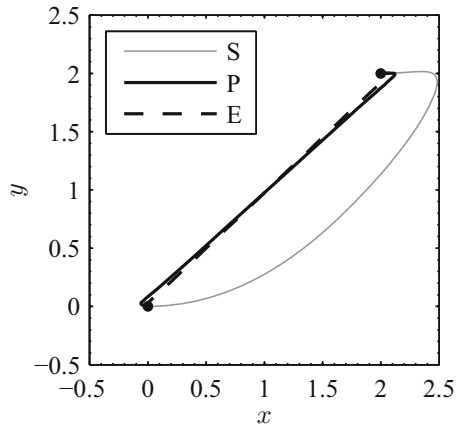

Fig. 6 Motion planning with state variable value minimization—path in XY plane

of the secondary task error ${}^1e(\vartheta)$ for both multiple task algorithms is of the same order of magnitude; however, for the egalitarian algorithm, the value of ${}^1e(\vartheta)$ reaches a lower value than for the prioritarian algorithm. The convergences of the main task error ${}^0e(\vartheta)$ for both multiple task algorithms are comparable. The value of the main task error ${}^0e(\vartheta)$ reached by the single task algorithm shows that the additional task influences the minimum value of the main task error reached by both multiple task algorithms. As was already said, also here the saturation and the fluctuation of the plots come from the inaccuracy of numeric computations.

5.2.3 Motion planning with obstacle/singularity avoidance

The last simulation shows a result of the motion planning composed with the avoidance of an obstacle. To determine this problem, it is necessary to define the obstacle function $h(y)$. This function $h(y)$ should have small values in the obstacle free regions, and take high values where the obstacles exist. Such a definition of the obstacle function provides that the algorithm finds control functions producing system trajectory which is

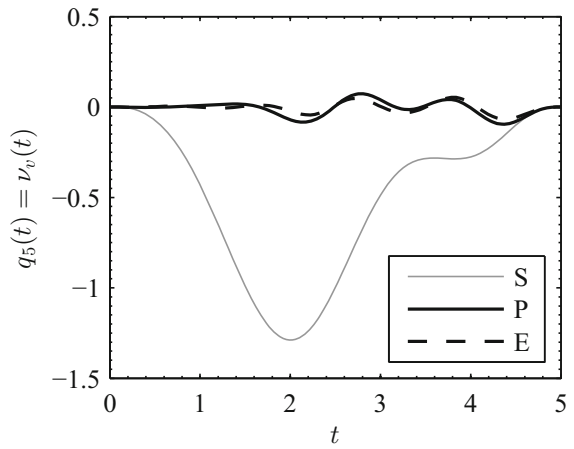

Fig. 7 Motion planning with state variable value minimization—trajectory of $q_5(t) = \nu_v(t)$

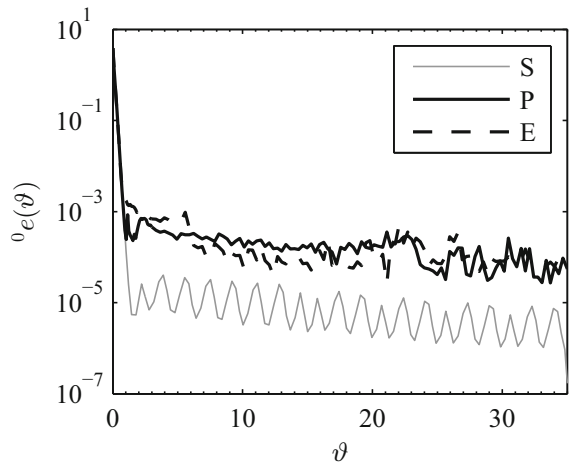

Fig. 8 Motion planning with state variable value minimization—convergence of ${}^0e(\vartheta)$

repelled from the obstacle. The definition of the obstacle function $h(y)$ refers to the potential field theory. In the presented case, the obstacle function was selected as

$$h(y) = \sum_i \frac{m_i}{\|y - y_{o_i}\|^2} + \left\| \exp((y - y_c)^2 - (y_d/2)^2) \right\|^2, \tag{39}$$

where operation y^2 means the element-wise power of vector y, $i = 1, 2, 3$ is the number of point-type obstacles, determined by the mass (radius) $m_i = 10$ and the position $y_{o_i} = \{(1, 1); (1, 4); (4, 1)\}$, $y_c = (2.5, 2.5)$ describes the center of the restrictive rectangle and

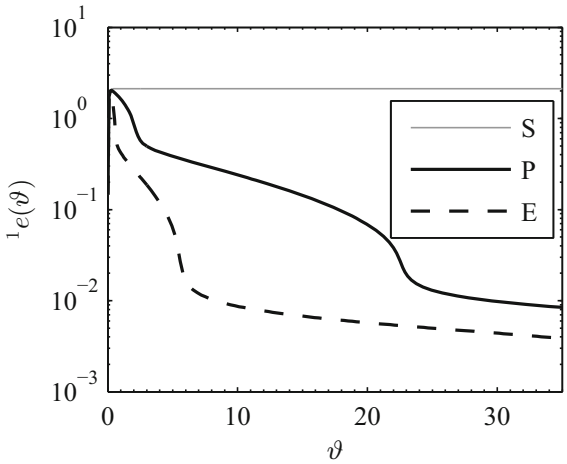

Fig. 9 Motion planning with state variable value minimization—convergence of $^1e(\vartheta)$

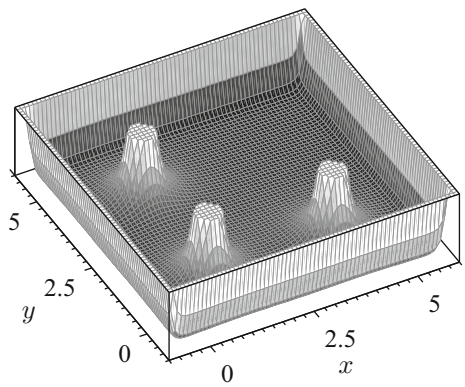

Fig. 10 Motion planning with obstacle avoidance—obstacles location

Table 4 Motion planning with obstacle avoidance—simulation parameters

q_0	q_d	T	$(^0\gamma, ^1\gamma)$	E	$u_0(t)$
$\begin{pmatrix}0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}5\\5\\0\\0\\0\\0\end{pmatrix}$	5	(10, 3)	$\begin{bmatrix} I_6 & 0 \\ 0 & 0.07 \end{bmatrix}$	$\begin{pmatrix}\exp(-t)\\ \exp(-t)\end{pmatrix}$

$y_d = \{3, 3\}$ represents the lengths of the rectangle edges, so in this case it is a square. Figure 10 presents the plot of the obstacle function $h(y)$. The remaining simulation parameters are collected in Table 4.

The results of the simulation are depicted in Figs. 11, 12, 13 and 14. The path in XY plane, together with the contour of $h(y)$ function is presented in Fig. 11. One can see that both algorithms, egalitarian and prioritarian, solve the planning problem. The resultant path reaches the desired point and avoids the obstacles. For comparison, we show that the trajectory obtained from the single task algorithm passes through an obstacle. The efficiency of the obstacle avoidance can be also observed in Fig. 12, where are depicted the values of the obstacle functions $h(y(t))$ along the trajectories of all three algorithms. The algorithms convergences are plotted in Figs. 13 and 14. Figure 14 shows that the prioritarian algorithm, for small values of ϑ, allows the error $^1e(\vartheta)$ to increase, while the main task error $^0e(\vartheta)$ decreases. On the contrary, in the egalitarian approach both errors decrease simultaneously. Another observation is that for the egalitarian algorithm the presence of

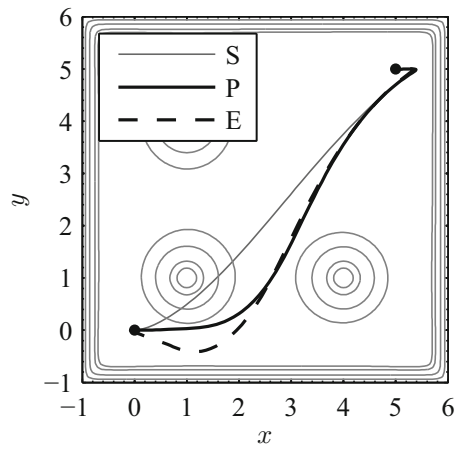

Fig. 11 Motion planning with obstacle avoidance—path in XY plane

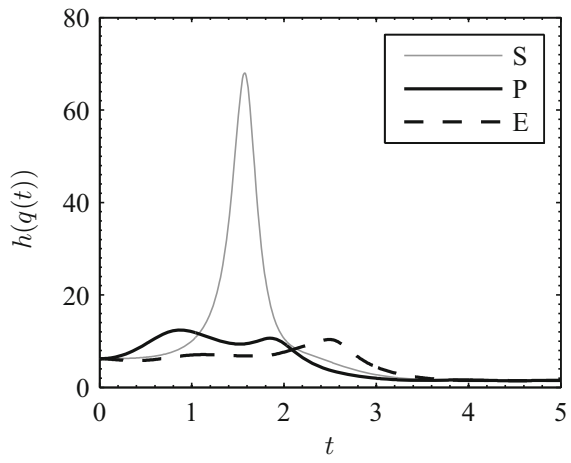

Fig. 12 Motion planning with obstacle avoidance—function $h(q(t))$ over the motion trajectory

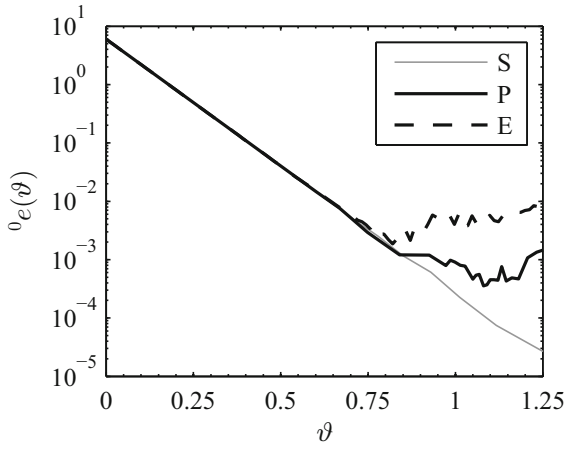

Fig. 13 Motion planning with obstacle avoidance—convergence of $^0e(\vartheta)$

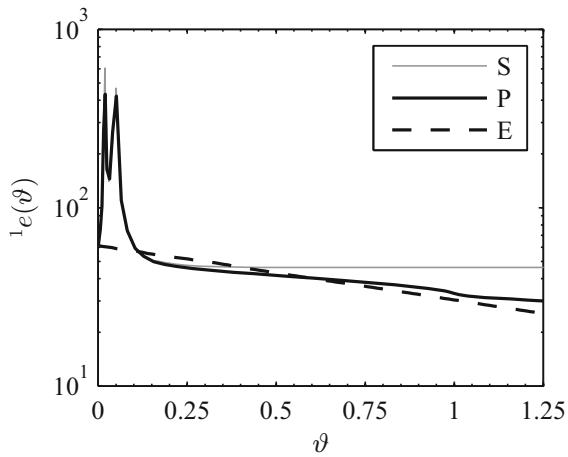

Fig. 14 Motion planning with obstacle avoidance—convergence of $^1e(\vartheta)$

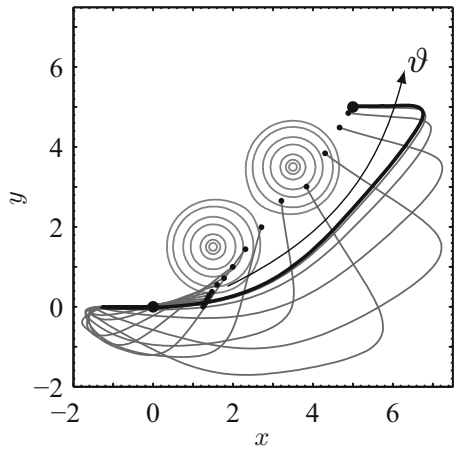

Fig. 15 Algorithm comparison—egalitarian algorithm solutions over ascending ϑ

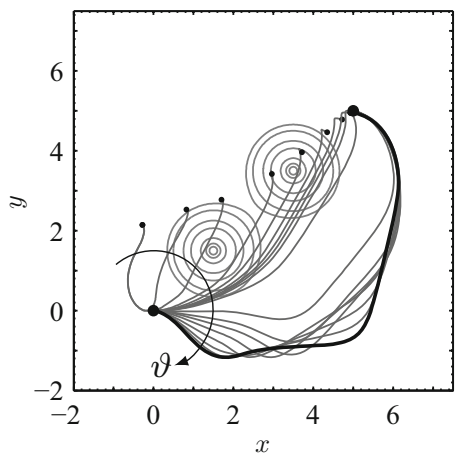

Fig. 16 Algorithm comparison—prioritarian algorithm solutions over ascending ϑ

the additional task affects the proper motion planning task solution. It can be observed that the final value of $^0e(\vartheta)$ is greater for the egalitarian than the prioritarian algorithm. Again, the fluctuations of the error convergences could be refined by increasing the computation accuracy, however such a modification may have impact on the computation time.

5.3 Algorithms efficiency comparison

In this section, we shall make the efficiency comparison between the egalitarian and the prioritarian algorithms. From the quantitative point of view, the egalitarian algorithm usually reaches smaller error values for the additional subtasks than the prioritarian algorithm. However, this fact usually affects the value of main task error which is often higher than in the prioritarian approach. From the qualitative point of view the main difference between the egalitarian and the proritarian algorithms is in the way of reaching the solution. We shall explain the differences by reference to Figs. 15 and 16, where the solution of the obstacle avoidance problem is presented for another obstacles placement.

With increasing ϑ parameter the egalitarian approach (Fig. 15) provides trajectories which simultaneously reach the desired point and guarantee the obstacle avoidance. The solution of the prioritarian algorithm

Table 5 The number of steps and the computation time of one step for each algorithm

	Energy minimization	State variable value minimization	Obstacle avoidance
S			
# steps	57	136	35
step time	0.234	0.234	0.234
P			
# steps	90	362	58
step time	0.439	0.610	0.387
E			
# steps	414	472	76
step time	0.336	0.486	0.353

(Fig. 16) very quickly (for small values of ϑ) reaches the destination point and then with increasing ϑ, begins to modify the trajectory in order to solve the additional subtask. Very often, the prioritarian approach allows the additional subtask error to increase (for small values of ϑ) as long as the main task error decreases.

Table 5 collects the number of steps necessary to solve the previously presented problems together with the time needed to compute single step of the particular algorithm. One can observe, that the prioritarian approach always needs more time to compute the single step than the egalitarian. This is due to the prioritarian algorithm is more difficult in implementation. On the other hand, as it can be seen in Table 5, the egalitarian approach requires more number of steps to obtain the solution than the prioritarian algorithm.

To sum up, from the quantitative point of view, both the algorithms return similar error values, so the choice of the right algorithm should be done by taking into consideration the qualitative point of view. If the two or more task have to be treated with the same weight then the egalitarian algorithm should be used. The prioritarian approach will have better efficiency for problems where the additional subtasks are essentially less important than the main task. The differences between the egalitarian algorithm and the prioritarian algorithm are important when there is not enough time to make long-term computations. As long as the evaluation time of one algorithm step for the egalitarian algorithm, as well as for the prioritarian algorithm is comparable, we have to decide if the computation could be stopped early. Let us assume that we cannot wait until ϑ reach large values. If someone chooses the egalitarian algorithm then for some small values of ϑ the system output $y(T)$ (proper motion planning) is still far from the desired point, obviously is should be closer to the desired point than the initial point. Also, the additional subtask is solved simultaneously, and finally, after a short computation time, one can obtain some temporary results where the system output is closer to the desired point and for example, the obstacles are avoided. If we repeat the above line of reasoning for the prioritarian algorithm then after a short-term computation the system output $y(T)$ could be much closer to desired point than it was in the egalitarian case. Nevertheless, the solution of the additional subtask could be poor, and the value of the additional subtask error could be even greater than it was for the initial control $u_0(t)$.

6 Conclusion

This paper has presented two different approaches to the multiple task motion planning problem for nonholonomic systems, which consists of the proper motion planning problem and one or more additional subtasks.

The equation of the egalitarian algorithm as well as of the prioritarian algorithm derived with the endogenous configuration space approach are functional differential equations. This work has shown a result of the multiple task algorithm computations involving the nonparametric version of the algorithms and the higher order, variable step-size differential equation solver. Such a numerical approach is characterized on a greater accuracy than the parametric and fixed-step method.

Both algorithms, the egalitarian and the prioritarian, have successfully solved three exemplary multiple task problems. The algorithms efficiency comparison has been done with respect to the control energy minimization, the state variable value minimization and the obstacle avoidance.

It is worth to mention that in some situation the prioritarian algorithm could provide the solution which does not solve all the subtasks. In such case the prioritarian approach leads to best possible solution. On the other hand, the egalitarian algorithm in difficult cases may not return any solution.

The accuracy of the computation could be tuned with built-in MATLAB functionality. However, the increase in computational accuracy results in increasing the computational time. The time needed by a 3.2 GHz

processor for one evaluation of the outer differential equation solver (i.e., for a particular value of ϑ) in the presented simulations is almost always below 0.5 s.

The presented subtasks constitute a base for the creation of more complicated subtasks. We have shown how to involve the control trajectory $u(t)$, the state space trajectory $q(t)$ and the output trajectory $y(t)$ in the additional subtask. In this paper we solve the problems with only two subtasks: the proper motion planning and one additional subtask. Both algorithms, the egalitarian and the prioritarian, can be easily expanded to solve problems with more than two subtasks.

Acknowledgements The author is deeply indebted to prof. Krzysztof Tchoń for providing many constructive suggestions for improving the text and to anonymous reviewers whose remarks allowed him to improve this paper. This research was supported by the National Science Centre, Poland, under the Grant Decision No DEC-2013/09/B/ST7/02368.

References

1. Antonelli, G.: Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. In: IEEE International Conference on Robotics and Automation, 2008. ICRA 2008, pp. 1993–1998 (2008)
2. Baillieul, J.: Kinematic programming alternatives for redundant manipulators. In: 1985 IEEE International Conference on Robotics and Automation. Proceedings, vol. **2**, pp. 722–728 (1985)
3. Ben-Israel, A., Greville, T.N.: Generalized inverses: theory and applications CMS books in mathematics. Springer, New York (2003)
4. Chang, P.: A closed-form solution for inverse kinematics of robot manipulators with redundancy. IEEE J. Robot. Autom. **3**(5), 393–403 (1987)
5. Chiaverini, S.: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Trans. Robot. Autom. **13**(3), 398–410 (1997)
6. Chiaverini, S., Oriolo, G., Walker, I.D.: Springer handbook of robotics, chap kinematically redundant manipulators. Springer, Berlin (2008)
7. Elmokadem, T., Zribi, M., Youcef-Toumi, K.: Trajectory tracking sliding mode control of underactuated AUVs. Nonlinear Dyn. **84**(2), 1079–1091 (2016)
8. Fantoni, I., Lozano, R.: Non-linear control for underactuated mechanical systems. Springer, London (2002)
9. Janiak, M., Tchoń, K.: Towards constrained motion planning of mobile manipulators. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 4990–4995 (2010)
10. Khaled, N., Chalhoub, N.G.: A self-tuning guidance and control system for marine surface vessels. Nonlinear Dyn. **73**(1), 897–906 (2013)
11. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. Int. J. Robot. Res. **6**(2), 3–15 (1987)
12. Pettersen, K.Y., Egeland, O.: Exponential stabilization of an underactuated surface vessel. In: 35th IEEE Conference on Decision and Control in Proceedings. Kobe, Japan, pp. 967–971 (1996)
13. Ratajczak, A., Karpińska, J., Tchoń, K.: Task-priority motion planning of underactuated systems: an endogenous configuration space approach. Robotica **28**, 885–892 (2010)
14. Ratajczak, A., Tchoń, K.: Multiple-task motion planning of non-holonomic systems with dynamics. Mech. Sci. **4**(1), 153–166 (2013)
15. Ratajczak, A., Tchoń, K.: Parametric and non-parametric Jacobian motion planning for non-holonomic robotic systems. J. Intell. Robot. Syst. **77**(3), 445–456 (2015)
16. Sciavicco, L., Siciliano, B.: A solution algorithm to the inverse kinematic problem for redundant manipulators. IEEE J. Robot. Autom. **4**(4), 403–410 (1988)
17. Sontag, E.D.: Mathematical control theory: deterministic finite dimensional systems, 2nd edn. Springer, New York (1998)
18. Sussmann, H.J.: A continuation method for nonholonomic path-finding problems. In: Proceedings of the 32nd IEEE Conference on Decision and Control, 1993, vol. **3**, pp. 2718–2723 (1993)
19. Tchoń, K., Jakubiak, J.: Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms. Int. J. Control **76**(14), 1387–1419 (2003)
20. Tchoń, K., Karpińska, J., Janiak, M.: Approximation of Jacobian inverse kinematics algorithms. Int. J. Appl. Math. Comput. Sci. **19**(4), 519–531 (2009)
21. Zadarnowska, K.: Switched modeling and task-priority motion planning of wheeled mobile robots subject to slipping. J. Intell. Robot. Syst. (2016). doi:10.1007/s10846-016-0397-1